APPENDIX

G

INTERRUPT VECTORS

A+CORE 1.3

Most microcomputer activity is interrupt-driven. This means that the CPU performs a task in response to an interrupt. Hardware, software, or the CPU itself can initiate an interrupt. An interrupt initiated by the CPU is called an "exception," and most often is the result of a problem or error, such as when the CPU is instructed to divide by zero.

Hardware interrupts begin when the BIOS controlling the hardware device raises its IRQ line. The interrupt controller receives the IRQ and signals the CPU that a hardware interrupt has occurred. The CPU responds by signaling to the interrupt controller that it will process the interrupt. The interrupt controller or the hardware BIOS then sends an interrupt value to the CPU. This interrupt value is used by the CPU to point to a row in the interrupt vector table. This row contains the memory location of the request handler that will service the hardware device. Software can also initiate an interrupt, but does not need or use the IRQ lines to do so, because the software is already running when the interrupt is issued, and so already has the attention of the CPU.

The memory locations listed in the interrupt vector table and the BIOS and device drivers assigned to these memory locations are determined during booting, but can be changed later. Software can make changes to the interrupt vector table to create its own user software interrupts, which it can later use to gain access to hardware devices. This ability of software to alter the interrupt vector table is sometimes used by a virus to replicate itself. Whether the CPU, hardware, or software initiates the interrupt, the same interrupt values are used, which are listed in the table below. These functions for the interrupt values are not universal; there are some variations.

Interrupt Value	Function
0	Divide by zero (exception interrupt)
1	Single step
2	Nonmaskable interrupt
3	Break point instruction
4	Overflow
5	Print screen

A+CORE 1.3

Interrupt Value	Function
6	Invalid opcode
7	Math coprocessor not present
8	System timer interrupt (IRQ 0)
9	Keyboard data ready (IRQ 1)
A	Reserved (IRQ 2) cascade from slave interrupt controller
B,C	Hardware interrupt for serial communication
D	Parallel port hardware interrupt (IRQ 5, LPT 2)
E	Diskette controller hardware interrupt (IRQ 6)
F	Printer hardware interrupt (IRQ 7, LPT 1)
10	Software interrupt to use video
11	Equipment check call
12	Memory check call
13	Software interrupt to use hard drive
14	Software interrupt to use serial port
15	Not used
16	Keyboard software interrupt
17	Software interrupt to use printer
18	ROM basic entry call
19	Bootstrap loader
1A	Time of day call
1B	Get control on keyboard break
1C	Get control on timer interrupt
1D	Pointer to video initialization table
1E	Pointer to diskette parameter table
1F	Pointer to graphics character generator
20–3F	Used by DOS
40	Reserved
41	Fixed disk parameter table
46	Fixed disk parameter table
47–5F	Reserved
60–67	Reserved for user software interrupts
68–6F	Not used
70	Real time clock hardware interrupt
72	Reserved (IRQ 10)
73	Reserved (IRQ 11)
74	Reserved (IRQ 12)
76	Fixed disk controller (IRQ 14)
77	Reserved (IRQ 15)
80–F0	Used by BASIC programs and interpreter
F1–FF	Not used